Remarks

Claims 1-17 remain in this application. Claims 1, 2, 3, 4, 9, and 10 are hereby amended. No new matter is being added.

Claim Rejections--Section 101

Original claims 9-16 were rejected under 35 USC 101 as being directed towards software per se. Examiner states that these original apparatus claims fail to recite any hardware features.

In accordance with Examiner's comments, applicant has amended independent claims 9 and 10 so that they now recite the necessary hardware features for a computer, including a processor and memory system. Claims 11-16 depend from claim 10. Therefore, applicant respectfully submits that claims 9-16, as amended, now overcome this rejection.

Claim Rejections--Section 103

The original claims were rejected under 35 U.S.C. § 103 as being unpatentable over Carini in view of Mitchell. This rejection is respectfully traversed with respect to the claims as now amended.

1. A method of cross-file inlining during a compilation of a program, the method comprising **determining which files to open and close based on affinity weightings between the files,** wherein the affinity weightings depend on a number of potential inlines between the files.

(Emphasis added.)

10/699,067
Amendment and Response to Office Action

Atty. Docket No. 200312985-1
March 22, 2007

The claim limitation of "**determining which files to open and close based on affinity weightings between the files**" is discussed, for example, in the first paragraph of the Summary section on page 2 of the present application.

The affinity weighting aspect is further described, for example, on pages 40-41 in relation to FIG. 11. For convenience of reference, FIG. 11 and the pertinent description are reproduced below.
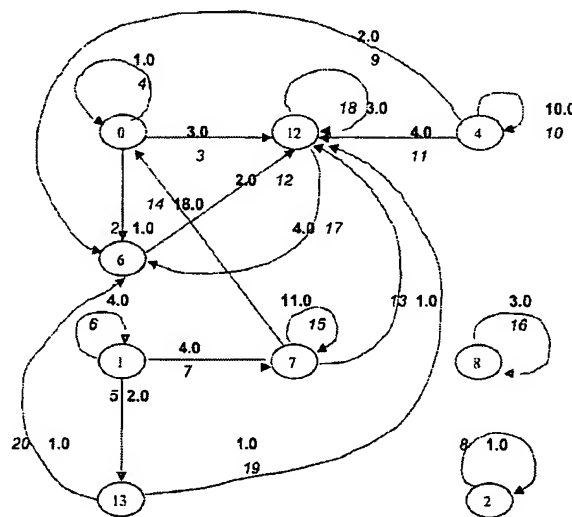

Figure 11

FIG. 11 shows the inline affinity graph that is obtained for a SPEC2000 integer benchmark, 164.gzip. There are 14 source files in this application. Based on the call sites selected for inlining by the analysis phase, 9 files are used by the inline transformation phase. **These files are numbered 0, 1, 2, 4, 6, 7, 8, 12, and 13. These are represented as nodes in the inline affinity graph as illustrated in FIG. 11.**

In FIG. 11, the edges are annotated with 2 values. The integer value (in italics) denotes the ID of the edge whereas the real value (in bold) denotes the weight of the edge. In the current implementation, **the weight denotes the number of call sites sharing the same relationship between caller and callee files.** A real value is used in order to facilitate easy incorporation of other factors in future implementations. For instance, let us consider the edges starting from node 0. There is a self-loop on node 0, numbered edge 4. The weight of 1.0 indicates that there is 1 call site whose caller and callee routines both reside in the same source file, namely file 0. There is an edge, 3, from source file, 0, to source file, 12. It has a weight of 3.0 indicating that there are 3 call sites where the caller routine resides in file 0 and the callee routine resides in file 12. It is to be noted that the caller and the callee routines are not necessarily the same for these 3 call sites, it is just that the files that they reside in are

10/699,067
Amendment and Response to Office Action

Atty. Docket No. 200312985-1
March 22, 2007

the same. There is an edge, 2, from source file, 0, to source file, 6. This edge has a weight of 1.0 indicating that there is a single call site with the caller routine residing in file 0 and the callee routine residing in file 6. The presence of the other edges in the inline affinity graph illustrated in FIG. 11 can be explained in a similar manner. While the inline affinity graph maintains the strength of the affinity between 2 source files, the actual call sites (or call-graph edges) are maintained separately as a list corresponding to every source file.

(Emphasis added.)

A procedure which uses the affinity weightings to determine which files to open is described, for example, on page 43, line 8, through page 44, line 24 of the specification, which is reproduced below for convenience of reference.

... The top-level algorithm is a simple iterative process consisting of the following broad steps:

- An edge from the inline affinity graph is chosen. **One embodiment of the invention simply chooses the edge with the highest weight since the weight signifies the number of inlines that can potentially be satisfied with the source files corresponding to the chosen edge.** One potential pitfall is the possibility of this edge having dependences on other edges of the inline affinity graph that have not yet been considered. However, in the general case, the percentage of inlines (for a certain edge in the inline affinity graph) that have dependences are assumed to be approximately uniform throughout the inline affinity graph. Assuming that the dependence relationship is not extremely biased towards a few ag-edges, this scheme should work well. In scenarios where the bias is extremely high towards a few ag-edges, this scheme will still work correctly but with reduced efficiency.
- **The source files corresponding to the chosen edge are retrieved.** It may be noted that these two source files may be the same signifying that the inlines are intra-file i.e. the caller and the callee reside in the same file.
- **Before inline transformation can proceed, the source files corresponding to the inline need to be opened.** This is referred to as the preparation phase for the impending inline.
- The preparation phase returns a status which can be one of the following values: prepn_true (i.e. preparation was successful and the inline can proceed), prepn_false (i.e. preparation was not successful), prepn_out_of_memory (i.e. preparation was not successful because a potential out_of_memory situation could be reached), and prepn_cold (preparation was not completed since the source files are cold).
- If the status returned by the preparation phase is prepn_true, it means that the associated source files could be opened and the inlines can proceed. If the status is prepn_out_of_memory, it means that a hold state has been reached. Some files need to be closed before this inline can proceed. In this case, StatusHandler is called. If the status returned by the preparation phase is prepn_cold, it means that the associated source files are considered cold and hence will not be opened at this point. In this scenario, the next edge in the inline affinity graph is examined.

- • StatusHandler is called when inlines cannot proceed in the current situation. StatusHandler usually **closes some files and opens others in order to allow inlines to proceed.** This routine is described in more detail later.

- After every inline, the affinity graph and the dependence graph are updated. The inline affinity graph is updated by reducing the weight of the corresponding edge. The dependence graph is implicitly updated. This means that the dependence test from edge x to edge y always checks first whether y is already inlined. If yes, there is effectively no dependence from edge x to edge y.

- The iterative process continues while there are remaining inlines to be performed. When the weights of all the edges in the inline affinity graph reach zero, all inlines have been done.

(Emphasis added.)

As described above, the affinity weightings are utilized to determine which files to open and close.

Applicant respectfully submits that the claim limitation pertaining to affinity weightings is <u>not</u> taught in either Carini, or Mitchell, or the combination thereof.

Regarding Carini, applicant agrees with the Examiner's assertion in the office action that Carini fails to disclose an inline affinity graph.

Regarding Mitchell, applicant respectfully submits that the description of the intermediate representation (IR) format in Mitchell does <u>not</u> teach **the use of affinity weightings to determine which files to open and close**, as required by amended claim 1. Regarding the IR format, various sections (col. 2: 42-49; col. 9: 42-49; and col. 10:46-47, 57-58, 61-64.) of Mitchell were cited in the office action. The cited sections of Mitchell are reproduced below for convenience of reference.

The intermediate representation format can accommodate annotations to avoid creation of auxiliary data structures. Such annotations can be used to thread graphs through the intermediate representation format. For example, data flow and control flow can be explicitly represented in the intermediate representation without creating a separate data structure. (Col. 2: 42-49.)

In any of the examples described herein, such auxiliary data structures can be implemented via the IR format by annotating the IR without having to construct a separate data structure. Separate data structures may be desired in some instances, but are not necessary to explicitly represent many auxiliary data structures. For example,

10/699,067
Amendment and Response to Office Action

Atty. Docket No. 200312985-1
March 22, 2007

a graph can be threaded through nodes of the IR rather than creating a separate data structure. (Col. 9: 42-49.)

To expressly represent control flow, links are made from branches in the software to destinations of the branch. For example, a link is made from the possible branch of the BNE instruction 920 from the operand 926 to the corresponding label 935 of the related label instruction 930 (e.g., the "TRUE" case). Another link is made from the possible branch of the BNE instruction 920 from the operand 927 to the corresponding label 955 (e.g., the "FALSE" case). In the example as in any of the examples herein, a label can be created for both cases.

Although the example shows a one-way link, the link can go the other way or both ways. Further, if more than one path to a label is possible, multiple links can be represented. "DO," "WHILE," "FOR," and other looping structures can also be so represented.

As shown in the example, control flow can be explicitly represented and expressed in the intermediate representation without modifying the format of the IR. Instead, the control flow is threaded through the IR. Other types of annotations can be combined with the ones shown as desired. (Col. 10: 46-64.)

Applicant respectfully submits that none of the above cited sections of Mitchell teaches **the use of affinity weightings to determine which files to open and close**, as required by amended claim 1.

Therefore, neither Carini, nor Mitchell, nor the combination thereof teaches the claimed invention per amended claim 1. Hence, applicant respectfully submits that amended claim 1 now overcomes this rejection.

Similarly, amended claim 2 now recites "**determining which files to open and close in dependence on affinity weightings between the files**." Therefore, for similar reasons as discussed above in relation to claim 1, claim 2 is also now patentably distinguished over the cited art.

Claims 3-8 depend from claim 2. Therefore, for at least the reasons as discussed above in relation to claim 2, claims 3-8 are also patentably distinguished over the cited art.

Claim 9 is an apparatus claim which is amended similarly to method claim 1. Therefore, for similar reasons as discussed above in relation to claim 1, claim 9 is also now patentably distinguished over the cited art.

10/699,067
Amendment and Response to Office Action

Atty. Docket No. 200312985-1
March 22, 2007

Claim 10 is an apparatus claim which is amended similarly to method claim 2. Therefore, for similar reasons as discussed above in relation to claim 2, claim 10 is also now patentably distinguished over the cited art.

Claims 11-16 depend from claim 10. Therefore, for at least the reasons as discussed above in relation to claim 10, claims 11-16 are also patentably distinguished over the cited art.

Similarly, original claim 17 recites "an inline transformer which performs function inlining within currently opened files and **determines which files to open and close in dependence of an affinity weighting between the files**." (Emphasis added.) Therefore, for similar reasons as discussed above, claim 17 is also now patentably distinguished over the cited art.

10/699,067
Amendment and Response to Office Action

Atty. Docket No. 200312985-1
March 22, 2007

Conclusion

For the above-discussed reasons, applicant respectfully submits that claims 1-17 are now in form for allowance. Favorable action is respectfully requested.
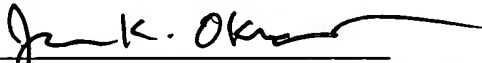
The Examiner is also invited to call the below-referenced attorney to discuss this case.
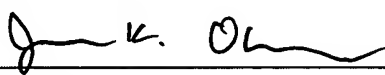
Respectfully Submitted,

Dhruva R. Chakrabarti

Dated: *March 22, 2007*

James K. Okamoto, Reg. No. 40,110
Tel: (408) 436-2111
Fax: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:	James K. Okamoto	Dated:	3/22/2007
Express Mail Mailing Number (optional):			